Hi Tech Ham Radio at http://www.amrad.org

The AMRAD Newsletter

The Amateur Radio Research and Development Corporation

Volume XXIII No. 4 July-August 1996

Software Radios
...a presentation by David Weinreich, WA2VUJ

Synopsis by Dick Barth, W3HWN

The speaker was formerly with COMSAT Laboratories, where he did some work on software radios, and is now with Globalstar, a Big (repeat: BIG) LEO company. A software radio is a receiver or transmitter in which some of the functions formerly performed using analog techniques are now done by digital signal processing (DSP) under software control. These functions include tuning, filtering, phase shifting, frequency conversion and demodulation. All these can be done digitally, if the digital circuitry is fast enough. The speaker described his "Ultimate Software Receiver", a wish-list box that would provide all the capabilities any ham could use. It would include an anti-aliasing (low-pass) filter, an A/D converter, some sort of DSP, a D/A converter and some filtering to get rid of clock noise, and a speaker. All this would be done under a controller which could be a PC or a more specialized machine. DSP can provide all sorts of demodulation, since this process can be represented as a mathematical function. It can also handle spectrum analysis and the identification and elimination of interference (selective filtration). The characteristics of the Ultimate Receiver were given as follows:

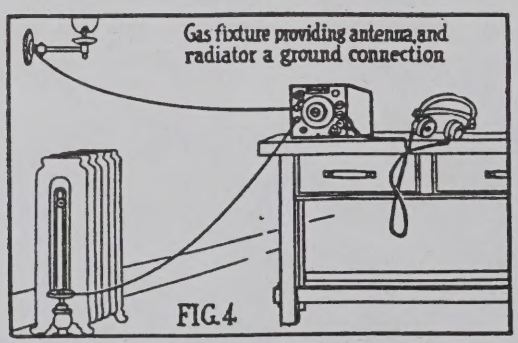
tuning range: 0-2 GHz
noise figure: 1 dB
maximum input power: 1 watt over a 30 MHz bandwidth
maximum input power: 0 dBW, or 20V pk-pk in 50 ohms
minimum input power (sensitivity): -206 dBW/Hz, 3 dB below noise

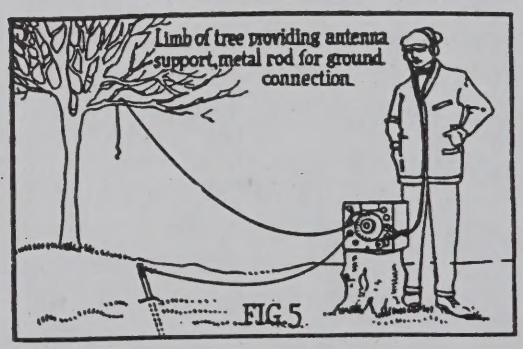
The sensitivity level given represents 1.94 uV RMS, or 6.72 uV per quantizing step, requiring an A/D converter with 22 to 24 bits of resolution. The sampling clock would be abour 2.5 times the maximum input frequency. The sampling width was chosen at about 1 picosecond. Aperture jitter was chosen as 0.019 femtoseconds; this measures the repeatability of the sampling time, and limits the signal to noise ratio. The DSP chip must sample at a 5 GHz rate, must be at least 22 bits wide, and must handle 500 million operations per second.

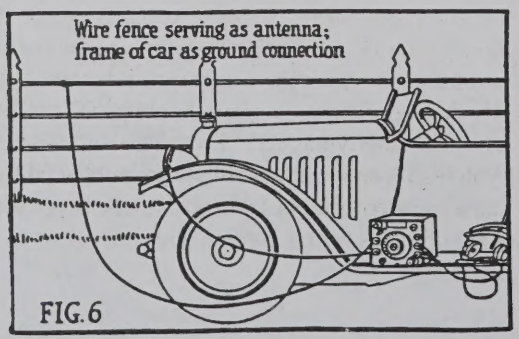
The "Ultimate" receiver is unfortunately outside the range of current technology, but may not be in a few more years. Using hardware available today, you can build a receiver using certain "shortcuts," such as undersampling the input signal. The Byquist criterion states that that sampling rate must be at least twice the highest signal bandwidth. A value of 2.5 is probablt the minimum practical because of the need for practical anti-aliasing filters. In other words, a realizable filter cannot be built that will cut off at precisely the highest input frequency, hence the sampling rate must be something more than twice that. To receive the entire 2 meter band, for example, a 10 MHz sampling rate would suffice.

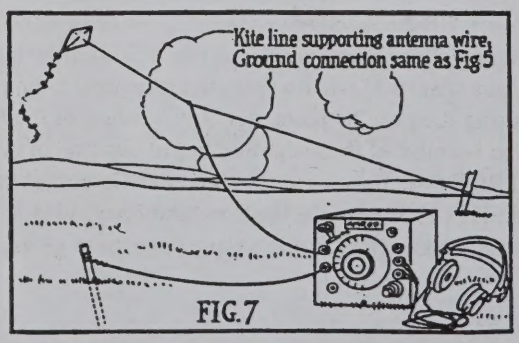
Existing hardware can provide 12 bits of resolution, 41 million operations per second, and 0-105 MHz input bandwidth and 5 picosecond aperture jitter. This is typical of what is found in DSP equipment today.

A digital receiver was designed including a divider, two double-balanced mixers and two direct digital synthesis chips by Analog Devices. these chips allow phase modulation, so can be programmed to provide a constant phase shift at the output. These components form the major part of the digital receiver, to which must be added the anti-aliasing filters, and an amplifier of some sort to precede the A/D converter.









An Introduction To Public Key Cryptography:

by André Kesteloot N4ICK.

In 1993, this Newsletter published two introductory articles of mine on Public Key Cryptography (PKC). Many readers and other interested parties have asked for reprints of these issues, which unfortunately are now out-of-print. (Remember that, *as usual*, you read it first in the AMRAD Newsletter). It was thus decided, particularly now that every other person you meet seems to be talking about PGP and such, to edit those two articles and reprint them into one.

Why should we, AMRAD members, be either concerned with, or even interested in Public Key Cryptography (PKC) ?

Firstly, we will soon find that PKC will play an increasingly important role in our daily life: Newspapers and magazines have started mentioning the likelihood of a "health-card" or even a "national identity card" as protection against health fraud and illegal immigration. These cards would almost certainly be "smart cards" protected from unauthorized snooping by various PKC schemes,

Secondly, as users of banking networks, we will use more and more often some PKC algorithms for authentication purposes (see "The Byzantine General's Problem" in the AMRAD Newsletter for Jan/Feb. 1993),

Thirdly, as engineers and/or programmers, we will find that access to our computers at the work-place will be regulated by smart cards, again using PKC,

And finally, as radio-amateurs, here is a domain where PKC should be particularly useful: the sending of authenticated commands to amateur-radio satellites. Indeed, the transmission of such commands is necessarily done over-the-air, hence can be intercepted. Assorted rascals could then try to spoof the satellite by replaying the same commands, if these were unprotected. (The use of encryption over-the-air is presently prohibited, and this specific application would obviously have to be first approved by the FCC).

Background:

Transferring information from point A to point B, while hiding its true meaning, has long been a cherished human endeavor. Cryptography was probably created the day after the first alphabet was invented, and since then, many clever schemes have been generated, only to be subsequently defeated. The traditional methods always relied on the two parties having a common reference. One side would then encode his / her message, using the common reference, and forward it to the intended recipient, who would then somehow look up the reference, and decode the message. (During the Civil War, for instance, it was not uncommon for people to own a Bible, and provided that both parties had agreed in advance to use a given chapter of a given book of the same edition of the Bible, then the message could be enciphered using simply the page number, the line number and the position of the desired word on that line.)

This being a fairly slow process, better machines were eventually introduced (read "The Code Breakers" for instance), and the introduction of computers certainly made things go faster, but essentially, it was still the same old problem: how to generate a better "key", and how to pass it so that both sides have access to it. Then came True Change with the introduction of Public Key Cryptography (I would venture to say that PKC will be seen, in retrospect, as one of the most momentous developments in technology during the past 20 years, not only because of the radically new approaches it offers to old problems, but also because of the sheer number of aspects of our lives it will quietly impact). I won't go into the history of PKC (you may want to reference "Contemporary Cryptology", an excellent anthology published by the IEEE in 1992) but the basic revolutionary idea which came to its creators in 1978 was to challenge the established axiom that the same key had to be available at both ends for encryption and decryption to be

able to take place. There are many different forms of PKC but the best known is probably the RSA system, named after its creators, Rivest, Shamir and Alderman. It uses one key for encryption (called the Public key because it can actually be made public) and another (Private or secret) key for decryption. The concept is based on the use of one-way, or trap-door, mathematical functions. These functions are mathematical operations trivial to perform in one direction, but extremely arduous in the reverse direction. For instance, to raise (23) to the third power, i.e.(23^3), we simply multiply 23*23*23 and instantly obtain 12,167. In the reverse direction though, extracting the cubic root of 12,167 is not quite so easy!

The old concept of cryptography usually evoked the idea of conspirators meeting in dark alleys for nefarious purposes; in fact every step of the exchange was somehow shrouded in secrecy. The new PKC system changed that too, and even the way an exchange is described in technical literature was modified: The token protagonists named in Public Key Cryptography scientific articles are no longer faceless entities, but have become plausible human beings, usually referred to as "Bob" and "Alice". Because this is all quite revolutionary, and yet impacts seriously on our everyday life, allow me to describe, step by step, a typical PKC transaction. (I have added Vader-the-Villain, a.k.a. Darth Vader, as the perennial person forever trying to intercept and understand the exchange of messages between Alice and Bob).

The New Process:

Let us suppose that Alice wants to send an encrypted message to Bob. Alice looks up Bob's public key in a publicly available directory. She uses it to encrypt her message and then posts this encrypted message in a public place. (This could be the local newspaper classified ads, or a banking network. Although the latter is certainly more private than the former, we will assume that it could be under attack by rogues, thieves and assorted cutthroats.) Now Bob retrieves the message, and decrypts it using his private- or secret key. This is the only step in the whole transaction that needs to be protected and kept confidential. Note that the public key and the private key are related to one another but that, if the one-way mathematical function is properly chosen, the private key cannot be derived from the public one. (Or, perhaps more accurately, can only be derived using exorbitant amounts of computing power, not usually available to your everyday villain in a convenient time frame. Note that if Bob has published several public keys, and Alice utilizes a new key for each transaction, then Vader-the-Villain is faced with even more serious problems!)

Key generation:

There are many ways to generate keys which are related to, but cannot easily be derived from, each other. Here is but one example. First select two large prime numbers (p) and (q), each possibly 100 digit long.

Their product $(n) = (p*q)$ *(Equation 1)*

will be a 200-digit long **public** encryption key. (There are certain restrictions in the choice of appropriate values for (p) and (q), which are beyond the scope of this overview, and the interested reader should look up the references listed at the end of this article).

The **private** or secret decryption key (d) is now obtained by calculating

$$d = [\, 2\,(p-1) * (q-1) + 1\,]/3 \qquad \textit{(Equation 2)}$$

Remember that the world-at-large, (including Alice and Vader-the-Villain) only knows Bob's public key (n), but if (n) is a 200-digit long number, it is very difficult, if not practically impossible to retrieve, with the sole knowledge of (n), the values of the two prime numbers (p) and (q).

Now to encrypt a message we can, for instance, take each letter or digit and convert it to its ASCII equivalent. Several of these ASCII values can now be grouped and encrypted by cubing them modulo (n). This will be the encrypted group (C) which will be transmitted by Alice to Bob.

Incidentally, a modulo operation is performed by dividing the number in question by the modulus, and posting only the remainder. Hence (4*4*4) mod 11 = 9 because 64/11=5 with a remainder of 9. Note that (15*15*15) mod 11 = 9; (26^3) mod 11=9; (37^3) mod 11=9; (48^3) mod 11=9, etc., which will make the job of Vader-the-Villain even more difficult, as the only information Alice sends to Bob is (C), the remainder of the modulo operation.

To decrypt the message (C), Bob must know his public encryption key (n) and his private/secret decryption key (d) and must calculate the result of [(C^d) mod n]; that is, Bob must raise (C) to the (d) power and then modulo (n) the result.

An Example:

Let us consider a practical example: in the January 1983 issue of Byte Magazine, John Smith published a very interesting article describing the implementation of a simplified form of the RSA algorithm. I have simplified it even further, and will present now two short BASIC programs you can use to demonstrate to yourself the concepts of PKC. To make my example both intelligible and manageable, I will use small number (but remember that the real system is only secure because it uses huge, 100-digit long prime numbers as factors.)

For the purpose of this simple demonstration, my two "large" prime numbers will be
(p) = 11 and (q) = 17.
Then, per Eq.1, (n) = (p*q) = 187
and, per Eq.2, (d) = [2(p-1)*(q-1)+1)/3 = 107

To summarize:

Bob's first prime number	(p) =	11
Bob's second prime number	(q) =	17
Bob's public encryption key	(n) =	187
Bob's private decryption key	(d) =	107

The only thing Alice knows is that Bob's public key is 187, and that she must cube her clear-text group and then "modulo 187" it. The result (C) will be transmitted to Bob, whose traffic, we will assume, could be somehow intercepted by Vader-the-Villain. Vader will thus know both (C) and (n) = 187 since this is Bob's public key.

Remember that, on the other hand, Bob knows both his public and his private keys. To decrypt Alice's message (C), Bob will have to calculate [(C^d) mod n].

Let's suppose that Alice wants to send Bob the letter "A" (her initial) whose ASCII equivalent is 65. Using Bob's public key (n) = 187, she calculates

(C) = [(65^3) mod 187] = [(65*65*65) mod 187] = 109 and she sends (109) to Bob.

Bob must now calculate [(109^107) mod 187] ! *Hmmmmm, say... what?* How specifically does Bob calculate the 107th power of 109? I thought you would never ask! There is a simple method to do so, described in some detail in the Byte 1983 article, and known as the "Russian Peasant's method." It is implemented in the short BASIC program listed below.

Anyway, amazingly enough [(109^107) mod 187] = (65), the number Alice originally wanted Bob to receive. Please note that Vader cannot derive the value of (65) from (187) and (109), the only two numbers he knows. Try it for yourself, remembering that the clear text you want to encrypt should be smaller than the public key you use.

BASIC Programs:

Here is the 6-line encryption program ENCRYPT.BAS

```
100 CLS'THIS IS A SIMPLE RSA ENCRYPTION PROGRAM, SEE BYTE JAN 83
110 INPUT "PLEASE ENTER THE PUBLIC KEY OF YOUR CORRESPONDENT";N
120 INPUT "NOW ENTER THE NUMBER YOU WANT TO ENCRYPT (0=END)";A
130 A1=A*A*A: A1=A1-INT(A1/N)*N'                    CUBE OF (A) THEN MODULO N
140 PRINT "THE ENCRYPTED BLOCK IS";A1:PRINT
150 IF A=0 THEN END ELSE GOTO 120'                  END, OR PLAY IT AGAIN SAM
```

Now for the corresponding decryption program DECRYPT.BAS

```
100 CLS'            THIS PROGRAM DECRYPTS RSA MESSAGES, SEE BYTE JAN 83.
110 DEFDBL C,D,M,N'                                 DOUBLE PRECISION
120 INPUT "PLEASE ENTER YOUR PUBLIC KEY";N
130 INPUT "AND NOW YOUR PRIVATE KEY";D
140 INPUT "AND NOW THE RECEIVED CRYPTOGRAM BLOCK (TO EXIT, ENTER 0)";C
150 IF C=0 THEN END'                                EXIT THIS PROGRAM
160 GOSUB 190'                                      START DECRYPTING
170 PRINT "AND THE CLEAR TEXT IS";M:PRINT
180 GOTO 140'                                       AND AGAIN
190 D1=D: M=1'                                      RUSSIAN PEASANT METHOD
200 IF D1/2=INT(D1/2) GOTO 220'                     SKIP IF D1 = EVEN
210 M=M*C: M=M-INT(M/N)*N'                          M = (M*C)MODULO N
220 C=C*C: C=C-INT(C/N)*N'                          C  = (C*C)MODULO N
230 D1=INT(D1/2): IF D1>0 GOTO 200
240 RETURN
```

How does this magick works? Let me now explain why and how the problem of efficiently raising a huge number to a huge power can best be tackled by a computer. (My approach is slightly different from the Russian Peasant Method originally described in the Byte article, and since I live in Virginia, I decided to name it the Northern Virginian Peasant Algorithm).

The Northern Peasant Algorithm:

Firstly, computers use the binary system, in which several operations can be performed with particular efficacy:

1). checking whether a number is even or odd is determined by looking at the least significant bit; if that bit = 0, the number is even, and if that bit = 1, the number is odd.

2). multiplying a number by 2, which simply means shifting everything one bit to the left and making the new least-significant bit = 0,

3). dividing an even number by 2, by removing the least significant bit and shifting everything one bit to the right,

4). checking whether a number is > 0, by checking the most significant bit.

Secondly, before we tackle the problem of raising a number X to a power Y, let us first examine how a computer can best be used to multiply efficiently a number X by another number Y, i.e., (X*Y). Let us open three registers x, y, and z and load them with the discrete values X, Y and 0 respectively.

Since we haven't modified anything, it is certainly true that (X*Y) = (x*y)+z (*Equation 3*)
Now for our process:

1). First check that Y > 0 , then

2). If Y is even, we can replace the value in register y by y/2 and the value in register x by 2x.

Again, we haven't changed the original relationship since we now have (2x*y/2)+z which still satisfies equation 3.

3). If Y is odd, then we can replace the value in register y by (y-1) and the value in register z by (z+x).

Equation 3 now becomes X*Y = [x*(y-1)+(z+x)] = [(x*y)-x+z+x] = [(x*y)+z] and our original relation still holds.

4). Now go back to (a) above and check again if Y is > 0. Keep performing the steps just described until Y=0, at which time, *stop!* The result of our operation is the value of register Z.

Example: to multiply 5 by 6, we will set the three registers x, y and z as follows: X=5, Y=6, Z=0

Is y=6>0 ? yes
Is 6 even ? yes, then replace y=6 by (6/2)=3 and replace x=5 by (5*2)=10
Is y=3>0 ? yes
Is y=3 even ? no, then replace y=3 by y=2 and z=0 by z=(0+10)
Is y=2>0 ? yes
Is y=2 even ? yes, then replace y=2 by y(2/2)=1 and x=10 by x=(10*2)=20
Is y=1 even ? no, then replace y=1 by y=0 and z=10 by z=z+x=(10+20)=30
Is y=0>0? no, then *stop!*
Result = (5*6) = value of register z = 30 (Answer)

Here is a short BASIC program I wrote to demonstrate the above Russian Peasant Method:

```
100 CLS:PRINT "MULTI.BAS, THE RUSSIAN PEASANT METHOD"
110 PRINT "TO MULTIPLY A NUMBER X BY ANOTHER NUMBER Y."
120 PRINT: INPUT "WHAT IS YOUR FIRST NUMBER (0=EXIT)";X
130 IF X=0 THEN END
140 INPUT "WHAT IS YOUR SECOND NUMBER ";Y
150 Z=0
160 IF Y>0 GOTO 170 ELSE GOTO 200
170 IF Y/2=INT(Y/2) GOTO 190'                    IS Y EVEN ?
180 Y=Y-1: Z=X+Z: GOTO 160'                      IF Y IS ODD
190 X=X*2: Y=Y/2: GOTO 160'                      IF Y IS EVEN
200 PRINT "THE ANSWER IS:";Z:PRINT:GOTO 120
```

Finally, here is the way to raise a number X to the Yth power. Again, let us create three registers x, y, and z and load them with the values X,Y and Z. (Note that now Z=1 and not 0 as in the above multiplication method.)

The relation between these three values is (X^Y)*Z *(Equation 4)*

Is Y>0 ? Then Y is either even or odd.
If Y is even, replace Y by Y/2,
Now replace X by X^2
Leave Z unmodified, and note that equation 4 still holds, as X^Y=(X^2)^Y/2*Z = (X^Y)*Z
Is Y odd? Then replace Y by Y-1
Leave X unmodified
Now replace Z by Z*X
Note that equation 4 still holds, i.e., X^Y=X^(Y-1)*ZX=(X^Y)*(X^ -1)*ZX=(X^Y)*Z
And go back to the first step above until Y=0

Example: to calculate 2^3, X=2 Y=3 Z=1

is y=3 >0 ? yes
is y=3 even? no, then replace y=3 by (y-1)=2 and replace z=1 by z=2
is y=2 even? yes, then replace y=2 by y=2/2=1 and replace x=2 by x^2=4
is y=1 even? no, then replace y=1 by y=1-1=0 and replace z=2 by z=2x=2*4=8
is y=0 >0 ? no, then *stop!*
Result= (2^3) = value of register Z = 8 (Answer)

I have modified the short BASIC program above to demonstrate what I have called the Northern Virginian Peasant Method:

```
100 CLS:PRINT "EXPON.BAS IS THE NORTHERN VIRGINIAN PEASANT"
110 PRINT "METHOD USED TO RAISE A NUMBER X TO A POWER Y."
120 PRINT:INPUT "WHAT IS YOUR FIRST NUMBER (0=EXIT)";X
130 IF X=0 THEN END
140 INPUT "NOW ENTER THE EXPONENT";Y
150 Z=1
160 IF Y>0 GOTO 170 ELSE GOTO 200
170 IF Y/2=INT(Y/2) GOTO 190'                    IS Y EVEN ?
180 Y=Y-1:Z=Z*X: GOTO 160'                       IF Y IS ODD
190 X=X*X: Y=Y/2: GOTO 160'                      IF Y IS EVEN
200 PRINT  "THE ANSWER IS:";Z:PRINT:GOTO 120
```

We have already established that the reason why these methods are efficient is due to the fact that they exploit the properties of binary arithmetic. One only needs to perform as many steps N as the power of 2 required to cover Y, i.e., $N = \log_2 Y$. Hence, since we always preserve the relationship in Eq.3 or Eq.4, these are not mathematical issues, but simply implementation ones.

Isn't that everything you always wanted to know as an introduction to Public Key and binary calculations?

Happy HamComputing
De André N4ICK

Bibliography:

- Beutelspacher, Albrecht. "Cryptology", The Mathematical Association of America, 1994.
- Halsall, Fred. "Data Communications, Computer Networks and Open Systems", Addison Wesley, London, 1992. pp. 596-599.
- Kahn, David "The Code Breakers", Macmillan, New York, 1967.
- Simmons, Gustavus, (Editor) "Contemporary Cryptology", IEEE Press, NY, 1992.
- Smith, John. "Public Key Cryptography", Byte Magazine, January 1983, pp. 198-216.
- Zimmerman, Philip R., "The Official PGP User's Guide", The MIT Press, Cambridge, Mass. 1995.

The AMRAD Page

The Summer of '96

This is being written during the dog days of summer. This weekend (August 17-18) started with a massive thunder and lightning storm on Friday afternoon. One of our stalwart members had the rare opportunity of flying alongside the leading edge of a violent storm, so close that thunder could be heard inside of the airplane. Lightning strikes could be seen blowing up trees and in general it was pretty exciting.

By Saturday morning, however, the weather was gone and the sun came up to shine on the annual microwave contest. We'll try to have a report in the next issue on how well the elite AMRAD microwave team did. Initial reports indicate that Murphy visited the audio section of at least one radio.

We can also use more members, in the DC area as well as around the world. We hear from people on the Internet a lot, including one recent note from South Africa.

Don't forget to join us on Saturdays for lunch at Tippy's Taco on Lee Highway just east of Nutley Street. Someone is always listening on 147.21 so give a call if you need directions.

Don't forget that a published article is worth a year's free membership!

Actual AMRAD MEMBERSHIP APPLICATION
Mail to AMRAD, Box 6148, McLean, VA 22106-6148
(but be sure to fill it out first!)

Annual Dues: Regular $20; 2nd in family at same address add $10; Canada and Mexico add $2; foreign surface mail add $2.30; foreign air mail add $8; if you live in the Metro DC area and use the very sophisticated wide area coverage AMRAD 2 meter repeater, please consider an additional donation for the upkeep of this equipment.

Name_____ Call _____ License class _____
Street Address _____
City _____ State/Province _____ Zip/Postal Code _____
Phone _____ (work) _____(home)
E-Mail address _____ ARRL member? _____

Interests __ deaf communications __RTTY/HF data __packet __spread spectrum
__DSP __satellites

I support the purposes of AMRAD.

(signed) _____

Everything you ever wanted to know about AMRAD. The Amateur Radio Research and Development Corporation (AMRAD) is a worldwide club of several hundred amateur radio and computer experimenters. AMRAD is incorporated in Virginia and is recognized by the U.S. Internal Revenue Service as a tax-exempt scientific and educational organization. The purpose of AMRAD is to develop skills and knowledge in radio and electronic technology, advocate design of experimental equipment and techniques, promote basic and applied research, organize technical forums and symposiums, collect and disseminate technical information, and provide experimental repeaters. AMRAD meetings are held on the second Thursday of each month at the Dolley Madison Library in McLean, VA. If the second Thursday is a holiday, an alternate date will be announced in the newsletter or on the AMRAD BBS. Except for the annual meeting in December, meetings are reserved for technical talks - not business. AMRAD is affiliated with the American Radio Relay League (ARRL), Foundation for Amateur Radio (FAR), Northern Virginia Radio Council (NOVARC), and the Mid-Atlantic Repeater Council (T-MARC). AMRAD airs "Newsline" every Sunday night at 8 p.m. on the WD4IWG repeater (147.21 MHz). Our Internet address is ramays@amrad.win.net. Our CompuServe address is 73156,2540. AMRAD has several computer bulletin board systems. The AMRAD Information Center, a real whiz-bang UNIX deal, is at (703) 849-1161. The Handicapped Education Exchange (HEX) is operated by Dick Barth, W3WHN. HEX has two numbers: (301) 593-7033 for TTYs and (301) 593-7357 for computer modems. AMRAD's WD4IWG repeater on 147.21 MHz is located at Tysons Corner, VA with remote receivers in Warrenton, VA and Washington, DC. Jeff Brennan, WB4WLW, is the repeater director. The AMRAD Newsletter (Copyright 1995 All Rights Reserved) is mailed six times a year to members and other clubs on an exchange basis. Technical articles, product announcements, news items, and other material related to amateur radio and computing are welcome. A year's free membership is given for original material accepted and published. Classified ads are free to members. Commercial ad rates are available (and very reasonable considering the level of impulse buying that goes on among our membership). The newsletter editor and publisher is Randy Mays, WA6VFC, who had to type all this stuff again when he switched to Microsoft Publisher. Many AMRAD members meet each Saturday afternoon at Tippy's Taco on Highway 29 near Cedar Lane just west of Merrifield, VA. Talk in, when someone remembers to turn on a radio, is on the 147.21 repeater. The AMRAD home page on the Internet is at http://www.amrad.org. There are some GREAT digital pictures on the Web site.

Amateur Radio Research
and Development Corporation
P.O. Drawer 6148
McLean, Virginia 22106-6148

957